**Министерство образования и науки РФ**

Федеральное государственное автономное образовательное учреждение высшего образования «Казанский (Приволжский) федеральный

университет»

ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА АНАЛИЗА ДАННЫХ И ИССЛЕДОВАНИЯ ОПЕРАЦИЙ Специальность (направление): 38.03.05 – Бизнес-информатика

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

(бакалаврская работа)

**МЕТОД РЕШЕНИЯ МНОГОКРИТЕРИАЛЬНЫХ ЗАДАЧ НА ОСНОВЕ МЕТРИКИ В ПРОСТРАНСТВЕ КРИТЕРИЕВ**

Работа завершена:

"\_\_\_"\_\_\_\_\_\_\_\_\_ 2018 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В.Вяльцева

Работа допущена к защите:

Научный руководитель

д-р физ.-мат. наук, профессор

"\_\_\_"\_\_\_\_\_\_\_\_\_ 2018 г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ И.Я.Заботин

Заведующий кафедрой

д-р физ.-мат. наук, профессор

"\_\_\_"\_\_\_\_\_\_\_\_\_ 2018 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_М.Д.Миссаров

**Казань - 2018**

**Содержание**

[Введение 3](#_Toc516399737)

[Глава 1. Теоретическая часть 5](#_Toc516399738)

[1.1. Метод на основе метрики в пространстве критериев 5](#_Toc516399739)

[1.2. Симплексный метод 7](#_Toc516399740)

[1.3. Метод внешних штрафных функций 10](#_Toc516399741)

[1.4. Метод наискорейшего спуска 12](#_Toc516399742)

[1.5. Метод дихотомии 13](#_Toc516399743)

[Глава 2. Практическая часть 15](#_Toc516399744)

[2.1. Описание работы программы 15](#_Toc516399745)

[2.2. Численные эксперименты 17](#_Toc516399746)

[2.3. Решение прикладной оптимизационной задачи 37](#_Toc516399747)

[Заключение 40](#_Toc516399748)

[Список используемой литературы 42](#_Toc516399749)

[Приложение. Листинг программы 43](#_Toc516399750)

# Введение

Во многих областях, от промышленного до социального сектора, лица, принимающие решения, сталкиваются с растущей необходимостью учитывать несколько критериев в процессе своей работы. К примеру, перед производителем может встать необходимость так организовать производственный процесс, чтобы получить максимальный объем выпуска продукции при минимальном браке. Часто для разрешения такого рода проблем можно составить математическую модель, представляющую собой задачу многокритериальной оптимизации. Неизвестные величины, которые подбираются в процессе решения задачи для получения оптимального решения, выбираются в качестве переменных. Они входят в целевые функции, представляющие собой рассматриваемые критерии или цели. Также вводятся ограничения на значения переменных. Например, может включаться ограничение на количество затрачиваемых ресурсов, которое не должно превышать их наличные запасы. Таким образом, ставится задача оптимизации нескольких функций на заданном допустимом множестве.

Для решения подобных задач существуют различные методы, которые находят компромиссные решения, учитывая противоречивый характер критериев и степень их важности. К основным подходам к решению многокритериальных задач относят метод последовательных уступок, метод свертки критериев, метод контрольных показателей, метод на основе метрики в пространстве критериев или, как его еще называют, метод «идеальной» точки, и др. В связи с частотой возникновения многокритериальных задач на практике исследование методов их решения по-прежнему актуально.

Данная выпускная квалификационная работа посвящена исследованию метода на основе метрики в пространстве критериев, сводящему исходную многокритериальную задачу к однокритериальной.

Целью выпускной работы является реализация программы, позволяющей находить решение многокритериальной задачи с линейными целевыми функциями и ограничениями, применяя один вариант метода на основе метрики в пространстве критериев и численное исследование этого метода. В качестве вспомогательного метода решения соответствующей однокритериальной задачи используется метод внешних штрафных функций. Для достижения поставленной цели необходимо было выполнить следующие работы:

1. Изучить подход к решению многокритериальных на основе метрики в пространстве критериев и метод внешних штрафных функций;
2. Программно реализовать исследуемый метод и спроектировать удобный пользовательский интерфейс;
3. Провести серию численных экспериментов для подтверждения работоспособности программы и проанализировать результаты;
4. Решить прикладную задачу многокритериальной оптимизации.

Для программной реализации метода используется язык C# и технология Windows Forms для создания пользовательского интерфейса.

Выпускная работа состоит из введения, двух глав, заключения, списка используемой литературы и приложения.

В первой главе содержатся теоретические основы исследуемого метода на основе метрики в пространстве критериев, выбираются и описываются подходы, которые применяются к решению вспомогательных задач, возникающих на различных этапах решения исходной задачи.

Во второй главе представлены результаты разработки программы. Показывается процесс работы и возможности пользовательского интерфейса, подтверждается работоспособность программы на серии примеров различных размерностей, решается прикладная задача многокритериальной оптимизации.

# Глава 1. Теоретическая часть

# 1.1. Метод на основе метрики в пространстве критериев

Пусть имеются частные оптимизационные критерии и множество ограничений . Каждый из частных критериев требуется максимизировать на допустимом множестве (задачи минимизации для упрощения записи сводятся к задачам максимизации ).

|  |  |  |
| --- | --- | --- |
|  |  | (1) |

Задача (1) является задачей многокритериальной оптимизации.

Положим . Тогда называют множеством абсолютных решений задачи многокритериальной оптимизации. Если , то в качестве решения задачи (1) естественно взять . Однако, как правило, . В этом случае в качестве решения многокритериальной задачи требуется найти такую точку , которая в определенном смысле была бы близка к каждому из множеств или значения близки к для .

Опишем метод на основе введения метрики в пространстве критериев, предлагающий способ выбора такой точки.

Рассматривается -мерное пространство оптимизационных критериев. Для вектор задает точку в этом пространстве.

Найдем вектор в пространстве критериев, состоящий из максимальных значений всех целевых функций, который назовем «идеальной» точкой:

.

В допустимом множестве практически никогда нет такой точки , что = . Если бы она была, то она принадлежала бы множеству и являлась бы решением задачи (1). Метод на основе метрики в пространстве критериев предлагает найти такую точку , что расстояние от до было бы минимальным. Для вычисления этого расстояния вводится некоторая метрика в -мерном пространстве критериев. Приведем примеры некоторых часто используемых мер расстояния. Манхэттенская метрика имеет следующий вид:

|  |  |  |
| --- | --- | --- |
|  |  | (2) |

Наиболее часто применяется евклидова метрика:

|  |  |  |
| --- | --- | --- |
|  |  | (3) |

Взвешенное евклидово расстояние используется, когда необходимо учитывать степень важности критериев. Каждому критерию приписывается некоторый вес

|  |  |  |
| --- | --- | --- |
|  |  | (4) |

Таким образом, после выбора метрики в качестве исходной принимают задачу:

|  |  |  |
| --- | --- | --- |
|  |  | (5) |

В данной выпускной работе рассматривается случай, когда все целевые функции и ограничения , задающие допустимое множество X, являются линейными функциями, и X – выпуклый многогранник. В качестве меры расстояния используется евклидово расстояние (3).

Целевые функции задаются следующим образом:

|  |  |  |
| --- | --- | --- |
|  |  | (6) |

А допустимая область:

|  |  |  |
| --- | --- | --- |
|  |  | (7) |

где

Оптимальное решение задачи с целевой функции и задачи с целевой функцией в виде ее подкоренного выражения совпадают, поэтому для упрощения расчетов в качестве решаемой задачи можно взять последнюю. Тогда целевая функция решаемой задачи будет являться выпуклой и гладкой и (8) будет представлять собой задачу выпуклой оптимизации.

|  |  |  |
| --- | --- | --- |
|  |  | (8) |

# 1.2. Симплексный метод

Для того, чтобы сформулировать задачу однокритериальной оптимизации (8), рассмотренной в предыдущем разделе, необходимо найти вектор оптимальных значений . Учитывая линейность частных критериев и функций, задающих ограничения, необходимо решить m задач линейного программирования. Для этого будем применять прямой симплексный метод с использованием метода дополнительных переменных и метода искусственного базиса. Для каждого -го критерия необходимо решить задачу:

|  |  |  |
| --- | --- | --- |
|  |  | (9) |

Если изначально задача была на нахождение минимума функции, то легко перейти к задаче максимизации: .

Симплекс-метод – это вычислительная процедура, которая основана на принципе последовательного улучшения планов - целенаправленном переборе допустимых базисов и соответствующих им опорных планов задачи, начиная с любого опорного плана. При переходе к следующему опорному плану значение целевой функции не убывает. Теоретически доказано, что если задача является невырожденной и оптимальное решение существует, то оно будет найдено за конечное число шагов.

Симплекс-метод применяется для задач, записанных в канонической форме:

1. Основные ограничения имеют вид равенств;
2. Для всех переменных задачи присутствуют условия их неотрицательности;
3. Свободные члены неотрицательны.

Если для некоторого -го уравнения свободный член окажется меньше нуля, обе его части умножают на -1.

Метод дополнительных переменных позволяет с помощью введения так называемых дополнительных переменных перейти от исходной задачи с ограничениями-неравенствами к новой задаче с ограничениями-равенствами. Они прибавляются к правым частям уравнений, если неравенство имело знак «≤», и вводятся в начальный базис, или отнимаются в случае «≥». Решение новой задачи дает решение исходной. Если все ограничения имели вид «≤», то после применения метода будет получен начальный базис.

Если в задаче есть переменные, на которые не наложено условие неотрицательности, то они заменяются разностью двух других неотрицательных переменных. Для произвольной переменной , где .

После проведения данных операций задача приведена к каноническому виду. Чтобы начать решать задачу симплексным методом, необходимо выбрать начальный базис и соответствующий ему опорный план, с которого начнется перебор. Если среди векторов-столбцов ограничений новой задачи нет единичного базиса, то применяется метод искусственного базиса.

Искусственные переменные вводятся в целевую функцию со знаком «» с большими положительными коэффициентами M. Если в оптимальном плане М-задачи не присутствуют искусственные переменные, то получено решение исходной задачи. Если же в оптимальном плане M-задачи хотя бы одна искусственная переменная будет отлична от нуля, то система ограничений исходной задачи линейного программирования (7) несовместна и задача (9) неразрешима.

Опишем сам симплексный метод.

Пусть . Базис представляет собой линейно независимую подсистему системы векторов-столбцов матрицы . Обозначим через – множество базисных значений индекса j, – базисная подматрица . Разложим векторы-столбцы по этому базису:

Вычислим значения , называемые оценками:

Введем симплексную таблицу , которая представляет собой матрицу , однозначно определяемую по некоторому базису .

,

где - вектор, составленный из базисных координат вектора), - вектор, составленный из базисных координат вектора

Симплекс-таблица содержит всю необходимую информацию для осуществления итерации симплекс-метода.

Сформулируем алгоритм метода:

1. Строится исходная симплексная таблица.
2. Проверяется критерий оптимальности: Если данное условие выполняется, то вектор , в котором и – решение задачи. Конец алгоритма.
3. Выбирается ведущий столбец k из условия: .
4. Строится .
5. Если , то задача не имеет решения в силу неограниченности целевой функции на допустимой области. Конец алгоритма.
6. Выбирается номер ведущей строки такой, что
7. Строится новый базис , где .
8. Пересчитывается симплексная таблица. Элементы , j=1...n ведущей строки делятся на ведущий элемент . Далее, осуществляя гауссовские преобразования, добиваются получения во всех остальных строках ведущего -ого столбца нулей. Рассчитываются новые оценки и . Возвращаемся к шагу 1.

# 1.3. Метод внешних штрафных функций

После нахождения вектора можно приступить к решению задачи однокритериальной оптимизации (8). Для этого применим один из методов внешних штрафных функций.

Методы штрафных функций относят к непрямым методам решения задач нелинейного программирования. В этой группе методов решение задачи условной оптимизации сводится к решению последовательности задач на безусловный экстремум. Методы внешних штрафных функций могут применяться для решения задач оптимизации как с ограничениями-неравенствами, так и с ограничениями-равенствами.

Опишем метод штрафных функций, используемый в данной работе.

Основная идея заключается в сведении исходной задачи

|  |  |  |
| --- | --- | --- |
|  |  | (10) |

к последовательности задач безусловной минимизации

|  |  |  |
| --- | --- | --- |
|  |  | (11) |

где – штрафная функция множества X.

Вспомогательная функция подбирается так, что она совпадает с на множестве X и быстро возрастает вне его. Можно ожидать, что ее быстрый рост вне X приведет к тому, что и при больших k нижняя грань этой функции на будет достигаться в точках, близких к множеству X, и решение поставленной задачи безусловной минимизации (11) приблизится к решению исходной задачи (10).

Последовательность функций , определенных и неотрицательных на , называется штрафной функцией множества X на множестве , если

Преобразуем множество (7) к следующему виду, перенеся свободные члены в правую часть уравнений и неравенств:

Выберем в качестве штрафной функции

|  |  |  |
| --- | --- | --- |
|  |  | (12) |

Так как все функции в рассматриваемом случае являются линейными, то и, соответственно, являются выпуклыми и непрерывно дифференцируемыми.

Рассматривается последовательность задач (11) со штрафными функциями (12). На шаге k с помощью какого-либо метода минимизации находят точку : , где – некоторая заданная последовательность, , k = 0,1,..., На следующем -м шаге точка используется в качестве начальной в используемом методе безусловной оптимизации. В качестве критерия остановки принимается .

К преимуществам метода штрафных функций относятся его простота и тот факт, что поиск минимума вспомогательной функции можно начинать из произвольной точки, т.е. нет необходимости искать точку из , как, например, в методах барьерных функций.

Основной недостаток метода – сложность функции , которая часто является овражной, степень ее овражности увеличивается с увеличением коэффициента . Сходимость многих методов безусловной минимизации для овражных функций очень медленная. Кроме того, промежуточные приближения не являются допустимыми решениями задачи (10), оптимум аппроксимируется извне области .

# 1.4. Метод наискорейшего спуска

Для решения последовательности задач (11) будем использовать метод наискорейшего спуска. Он применим к этим задачам, так как являются гладкими функциями, как было указано ранее.

Группа градиентных методов относится к методам первого порядка, т.е. на каждой итерации используются первые частные производные оптимизируемой функции . Для построения последовательности приближений в качестве направлений итерационного перехода выбираются антиградиенты функции, вычисленные в точках , т.е.

где – шаг градиентного метода.

Метод сходится, если или . При произвольном выборе шага, например, постоянного, последовательность может оказаться неминимизирующей и метод в этом случае расходится.

Метод наискорейшего спуска использует полный шаг при осуществлении переходов. Находится шаговый множитель любым методом одномерной оптимизации такой, что выполняется:

|  |  |  |
| --- | --- | --- |
|  |  | (13) |

Сходимость при таком выборе шага теоретически доказана для выпуклых и непрерывно дифференцируемых на функций при ограниченном множестве .

Вычисления продолжаются до тех пор, пока не выполнится критерий останова, например, .

Недостатком градиентных методов является то, что они малоэффективны для овражных функций, так как происходит спуск в овраг и блуждание от одного склона к другому без существенного продвижения по дну оврага.

# 1.5. Метод дихотомии

Для нахождения полного шага из условия (13) используется метод дихотомии или метод половинного деления. Он относится к методам прямого поиска, т.е. при поиске экстремума функции используются только ее вычисленные значения.

Предположим, что найден отрезок [a,b], который содержит точку минимума функции . Задаются , .

1. Положим i = 1.
2. Разобьём заданный отрезок пополам и возьмём две симметричные относительно центра точки и так, что:
3. Вычислим и сравним значения функции в двух новых точках. Если , то полагается В противном случае, .
4. Если , то = и алгоритм заканчивает свою работу. Иначе полагается и осуществляется переход к шагу 2.

Выбор начального отрезка [a,b], содержащего , можно провести следующим образом. Положим , - возрастающая последовательность. Начинаем с = 2.

Если на -ом шаге , то в качестве исходного отрезка выберем []. Иначе переходим к следующему шагу .

# Глава 2. Практическая часть

# 2.1. Описание работы программы

Для программной реализации метода используется язык C# и технология Windows Forms для создания пользовательского интерфейса.

В пользовательском интерфейсе программы предусмотрено два вида ввода данных: считывание с текстового файла и ввод данных вручную на форме. Для считывания задачи с файла должен быть соблюден определённый формат данных, описанный в справке, которую можно увидеть при наведении мыши на знак вопроса. Для того чтобы ввести данные вручную, необходимо указать число переменных, критериев и ограничений.

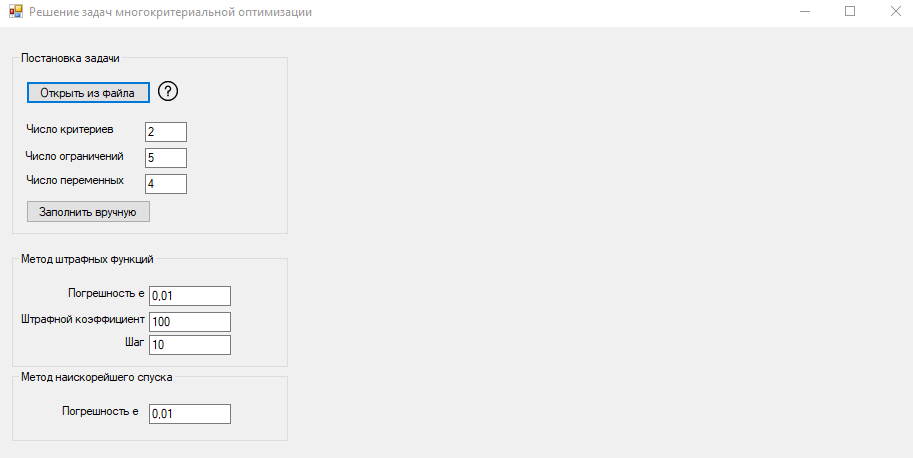


Рисунок 1 Главное окно при запуске программы

После выбора типа ввода начальных данных появляются выпадающие списки и таблицы, где можно скорректировать или ввести числовые значения. Есть возможность сохранить введенную задачу в формате, который считывает данная программа, нажав на кнопку «Сохранить задачу».

Осуществив постановку задачи многокритериальной оптимизации, пользователь может найти «идеальную» точку, нажав кнопку «Рассчитать вектор ». При этом выводится сам вектор, рассчитанный с помощью симплексного метода, и постановка соответствующей задачи однокритериальной оптимизации, получаемая при применении метода на основе метрики в пространстве критериев.

Данная задача решается методом внешних штрафных функций. Для него необходимо указать погрешность, с которой решается задача, начальный штрафной коэффициент и шаг, с которым он будет увеличивается. На каждом шаге метода решается задача безусловной минимизации методом наискорейшего спуска, для которого также необходимо задать погрешность.

Для получения решения необходимо нажать на кнопку «Рассчитать».

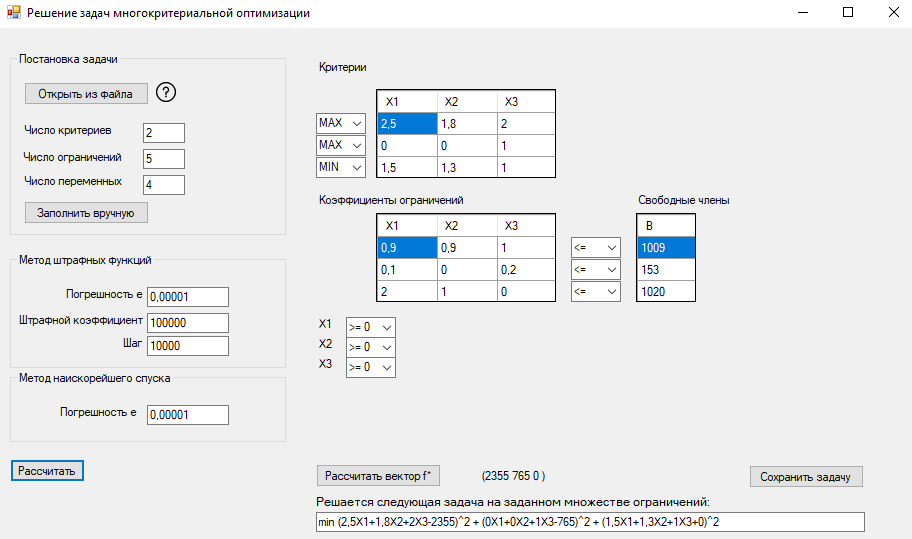


Рисунок 2 Главное окно с поставленной задачей

После осуществления расчетов выводится окно с решением задачи. Выводятся итерации метода штрафных функций, значение целевой функции на полученном решении, т.е. расстояние до «идеальной» точки, время работы программы и сумма невязок, показывающая удаленность решения от допустимой области.

Результаты можно сохранить в текстовый файл .txt или excel-файл формата .xlsx.

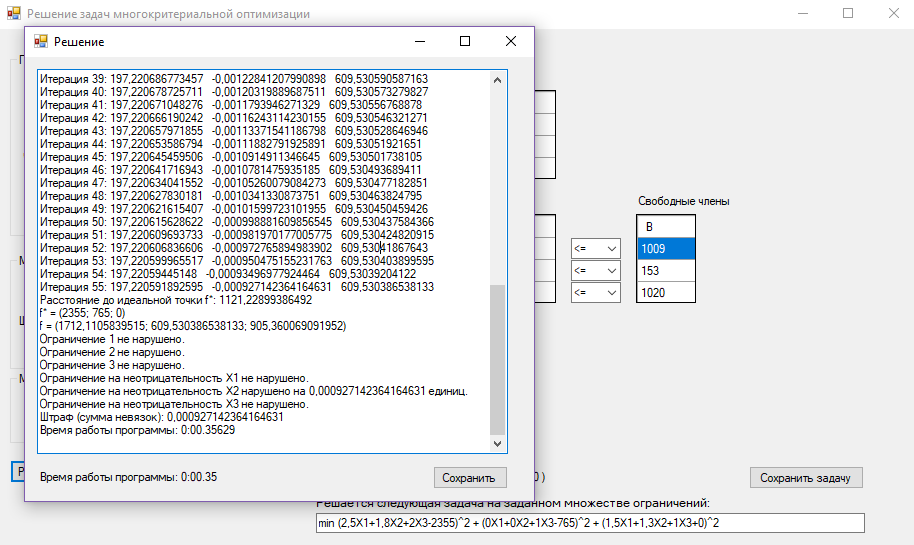


Рисунок 3 Окно с решением задачи

# 2.2. Численные эксперименты

Пример 1.

Найдем вектор оптимальных значений . Решим задачу линейного программирования с первым критерием.

Приведем задачу в каноническую форму, введя дополнительные переменные.

Построим симплексную таблицу.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| БАЗИС | CB | XB | -1 | 3 | 0 | 0 | 0 | 0 |
| x1 | x2 | x3 | x4 | x5 | x6 |
| 3 | 0 | 1 | -1 | 1 | 1 | 0 | 0 | 0 |
| 4 | 0 | 0 | 1 | -2 | 0 | 1 | 0 | 0 |
| 5 | 0 | 4 | 1 | 0 | 0 | 0 | 1 | 0 |
| 6 | 0 | 3 | 0 | 1 | 0 | 0 | 0 | 1 |
|  |  | 0 | 1 | -3 | 0 | 0 | 0 | 0 |

Найдена отрицательная оценка . Ведущим является второй столбец. . Найдем : min{1,3}=1. Первая строка - ведущая. Во множество базисных индексов B вместо 3 войдет 2. Пересчитаем симплексную таблицу с помощью правила прямоугольника.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| БАЗИС | CB | XB | -1 | 3 | 0 | 0 | 0 | 0 |
| x1 | x2 | x3 | x4 | x5 | x6 |
| 2 | 3 | 1 | -1 | 1 | 1 | 0 | 0 | 0 |
| 4 | 0 | 2 | -1 | 0 | 2 | 1 | 0 | 0 |
| 5 | 0 | 4 | 1 | 0 | 0 | 0 | 1 | 0 |
| 6 | 0 | 2 | 1 | 0 | -1 | 0 | 0 | 1 |
|  |  | 0 | -2 | 0 | 3 | 0 | 0 | 0 |

Найдена отрицательная оценка . Ведущим является первый столбец. . Найдем : min{4,2}=2. Четвертая строка - ведущая. Во множество базисных индексов B вместо 6 войдет 1. Пересчитаем симплексную таблицу с помощью правила прямоугольника.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| БАЗИС | CB | XB | -1 | 3 | 0 | 0 | 0 | 0 |
| x1 | x2 | x3 | x4 | x5 | x6 |
| 2 | 3 | 3 | 0 | 1 | 0 | 0 | 0 | 1 |
| 4 | 0 | 4 | 0 | 0 | 1 | 1 | 0 | 1 |
| 5 | 0 | 2 | 0 | 0 | 1 | 0 | 1 | -1 |
| 1 | -1 | 2 | 1 | 0 | -1 | 0 | 0 | 1 |
|  |  | 7 | 0 | 0 | 1 | 0 | 0 | 2 |

Нет отрицательных оценок. Оптимальный план: (2;3), значение целевой функции: 7.

Решим задачу линейного программирования со вторым критерием.

Приведем задачу в каноническую форму, введя дополнительные переменные.

Построим симплексную таблицу.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| БАЗИС | CB | XB | 4 | -1 | 0 | 0 | 0 | 0 |
| x1 | x2 | x3 | x4 | x5 | x6 |
| 3 | 0 | 1 | -1 | 1 | 1 | 0 | 0 | 0 |
| 4 | 0 | 0 | 1 | -2 | 0 | 1 | 0 | 0 |
| 5 | 0 | 4 | 1 | 0 | 0 | 0 | 1 | 0 |
| 6 | 0 | 3 | 0 | 1 | 0 | 0 | 0 | 1 |
|  |  | 0 | -4 | 1 | 0 | 0 | 0 | 0 |

Найдена отрицательная оценка . Ведущим является первый столбец. . Найдем : min{0,4}=0. Вторая строка - ведущая. Во множество базисных индексов B вместо 4 войдет 1. Пересчитаем симплексную таблицу с помощью правила прямоугольника.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| БАЗИС | CB | XB | 4 | -1 | 0 | 0 | 0 | 0 |
| x1 | x2 | x3 | x4 | x5 | x6 |
| 3 | 0 | 1 | 0 | -1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | -2 | 0 | 1 | 0 | 0 |
| 5 | 0 | 4 | 0 | 2 | 0 | -1 | 1 | 0 |
| 6 | 0 | 3 | 0 | 1 | 0 | 0 | 0 | 1 |
|  |  | 0 | 0 | -7 | 0 | 4 | 0 | 0 |

Найдена отрицательная оценка . Ведущим является второй столбец. . Найдем : min{2,3}=2. Третья строка - ведущая. Во множество базисных индексов B вместо 5 войдет 2. Пересчитаем симплексную таблицу с помощью правила прямоугольника.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| БАЗИС | CB | XB | 4 | -1 | 0 | 0 | 0 | 0 |
| x1 | x2 | x3 | x4 | x5 | x6 |
| 3 | 0 | 3 | 0 | 0 | 1 | 0.5 | 0.5 | 0 |
| 1 | 0 | 4 | 1 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 2 | 0 | 1 | 0 | -0.5 | 0.5 | 0 |
| 6 | 0 | 1 | 0 | 0 | 0 | 0.5 | -0.5 | 1 |
|  |  | 14 | 0 | 0 | 0 | 0.5 | 3.5 | 0 |

Нет отрицательных оценок. Оптимальный план: (4;2), значение целевой функции: 14.

Вектор найден.

Таким образом, задача однокритериальной оптимизации (8) имеет вид:

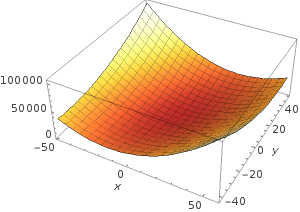


Рисунок 4 График функции

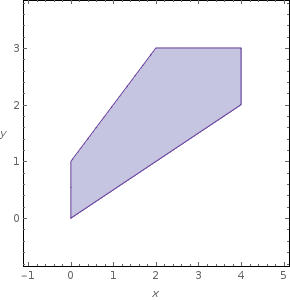


Рисунок 5 Область ограничений X

Точка безусловного минимума (4.4545; 3.8181), которая отображается в в пространстве критериев, лежит вне области X, решение задачи условной минимизации будет достигаться на границе X.

Применим метод штрафных функций и распишем одну его итерацию. Пусть .

В методе наискорейшего спуска положим начальную точку (0;0), .

1.,

2.,

3. ,

4. ,

5. ,

6. ,

7. ,

8. ,

9. ,

,

11. ,

На первой итерации метода штрафных функций получили ответ .

Приведем первые 20 итераций метода штрафных функций:

Итерация 1: 4,38004863781401 3,68977418435666

Итерация 2: 4,14313622071616 3,28603194247006

Итерация 3: 4,0861439965634 3,1808037672697

Итерация 4: 4,06124324309378 3,1323158701289

Итерация 5: 4,04791264719585 3,10421083360661

Итерация 6: 4,03849078568515 3,08628003233899

Итерация 7: 4,03297123836817 3,07332585479437

Итерация 8: 4,02991285819503 3,06326154031471

Итерация 9: 4,02427620040067 3,0569468072027

Итерация 10: 4,02199625477916 3,05100496521625

Итерация 11: 4,01975811510691 3,0462422062706

Итерация 12: 4,01800885652608 3,04234915751365

Итерация 13: 4,01651672899812 3,03901616062924

Итерация 14: 4,01527633896923 3,03622459310645

Итерация 15: 4,01420183127347 3,03378830490793

Итерация 16: 4,01326384278927 3,03165034311419

Итерация 17: 4,01242571400281 3,02973037184177

Итерация 18: 4,01171923502888 3,02810548369281

Итерация 19: 4,01107103104212 3,02660585408123

Итерация 20: 4,01048832775195 3,02525437355401

…

При уменьшении погрешности метод приближается к решению

.

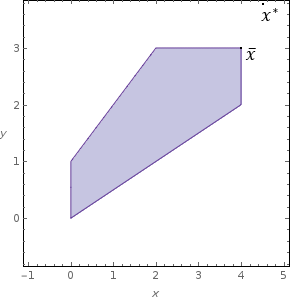


Рисунок 6 Решение , соответствующее «идеальной» точке , и найденное решение

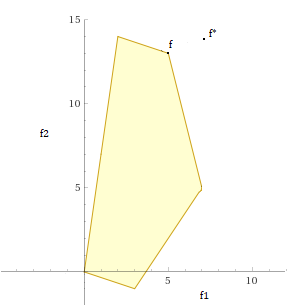


Рисунок 7 Точки в пространстве критериев

Расстояние между «идеальной» точкой и точкой, соответствующей полученному решению, :

Пример 2.

Итерация 88: 14,3621461235287 -9,06191851357141E-05 95,2309324672206

Расстояние до идеальной точки f\*: 129,088623304701

f\* = (1079,50617283951; 84,0091836734694)

f = (1022,68326977663; 199,918762027504)

Ограничение 1 не нарушено.

Ограничение 2 не нарушено.

Ограничение 3 не нарушено.

Ограничение на неотрицательность X1 не нарушено.

Ограничение на неотрицательность X2 нарушено на 9,06191851357141E-05 единиц.

Ограничение на неотрицательность X3 не нарушено.

Штраф (сумма невязок): 9,06191851357141E-05

Время работы программы: 0:00.0401

Пример 3.

Итерация 11: -0,000319170729758017 8,65622947907844 32,6698108661472

Расстояние до идеальной точки f\*: 83,0093650125355

f\* = (87,7049180327869; 438,439024390244)

f = (130,147656809194; 367,100615346153)

Ограничение 1 не нарушено.

Ограничение 2 не нарушено.

Ограничение 3 нарушено на 0,000255595878741133 единиц.

Ограничение на неотрицательность X1 нарушено на 0,000319170729758017 единиц.

Ограничение на неотрицательность X2 не нарушено.

Ограничение на неотрицательность X3 не нарушено.

Штраф (сумма невязок): 0,000574766608499149

Время работы программы: 0:08.91

Пример 4.

Итерация 15: 4,52598849163531 4,2626444546658 0,105089691127258 3,73634276558793

Расстояние до идеальной точки f\*: 66,2083500139378

f\* = (11,1111111111111; 3,48837209302326)

f = (20,3923987827444; 69,0429547758876)

Ограничение 1 нарушено на 2,73152435994461E-05 единиц.

Ограничение 2 не нарушено.

Ограничение 3 не нарушено.

Ограничение 4 не нарушено.

Ограничение на неотрицательность X1 не нарушено.

Ограничение на неотрицательность X2 не нарушено.

Ограничение на неотрицательность X3 не нарушено.

Ограничение на неотрицательность X4 не нарушено.

Штраф (сумма невязок): 2,73152435994461E-05

Время работы программы: 0:00.00602

Пример 5.

Итерация 3: 2,46057837752455 -3,61206264436309E-05 2,95599484853176 0,211279768753052 1,99740823028646

Расстояние до идеальной точки f\*: 179,783648230933

f\* = (219,222379854584; 4,35483870967742)

f = (41,06648279981; 28,4927034833574)

Ограничение 1 не нарушено.

Ограничение 2 не нарушено.

Ограничение 3 не нарушено.

Ограничение 4 не нарушено.

Ограничение 5 нарушено на 1,68918960241626E-06 единиц.

Ограничение на неотрицательность X1 не нарушено.

Ограничение на неотрицательность X2 нарушено на 3,61206264436309E-05 единиц.

Ограничение на неотрицательность X3 не нарушено.

Ограничение на неотрицательность X4 не нарушено.

Ограничение на неотрицательность X5 не нарушено.

Штраф (сумма невязок): 3,78098160460471E-05

Время работы программы: 0:16.15399

Пример 6.

Итерация 7: -1,26915082605523E-06 11,5552381689422 1,25731745748817 13,4566010007619 25,3532611718391 24,5523633104948

Расстояние до идеальной точки f\*: 2,72828757089788

f\* = (628,195596951736; 331,646062658764)

f = (626,085489620798; 333,375513473744)

Ограничение 1 не нарушено.

Ограничение 2 не нарушено.

Ограничение 3 не нарушено.

Ограничение 4 не нарушено.

Ограничение 5 не нарушено.

Ограничение на неотрицательность X1 нарушено на 1,26915082605523E-06 единиц.

Ограничение на неотрицательность X2 не нарушено.

Ограничение на неотрицательность X3 не нарушено.

Ограничение на неотрицательность X4 не нарушено.

Ограничение на неотрицательность X5 не нарушено.

Ограничение на неотрицательность X6 не нарушено.

Штраф (сумма невязок): 1,26915082605523E-06

Время работы программы: 0:00.32925

Пример 7.

Итерация 29: 35,3260731234187 13,2803799825556 -6,09561348001627E-05 53,5339668611619 27,2614750967526 26,725303868341 22,1052147896112

Расстояние до идеальной точки f\*: 455,672752302402

f\* = (1235; 211,25)

f = (951,074225496608; 567,654000768083)

Ограничение 1 не нарушено.

Ограничение 2 не нарушено.

Ограничение 3 не нарушено.

Ограничение 4 не нарушено.

Ограничение 5 не нарушено.

Ограничение на неотрицательность X1 не нарушено.

Ограничение на неотрицательность X2 не нарушено.

Ограничение на неотрицательность X3 нарушено на 6,09561348001627E-05 единиц.

Ограничение на неотрицательность X4 не нарушено.

Ограничение на неотрицательность X5 не нарушено.

Ограничение на неотрицательность X6 не нарушено.

Ограничение на неотрицательность X7 не нарушено.

Штраф (сумма невязок): 6,09561348001627E-05

Время работы программы: 0:27.24167

Пример 8.

Итерация 2: 1,1444645296443 -2,53623049360914E-05 1,15426741075667 1,18461754666253 0,982401586103252 1,97724335735864 0,864798739216008 1,82818326166846

Расстояние до идеальной точки f\*: 134,835323021472

f\* = (195,3; 87,4673366834171)

f = (70,7944089510066; 35,7090317281207)

Ограничение 1 не нарушено.

Ограничение 2 не нарушено.

Ограничение 3 нарушено на 4,87184228958881E-06 единиц.

Ограничение 4 не нарушено.

Ограничение 5 не нарушено.

Ограничение 6 не нарушено.

Ограничение 7 не нарушено.

Ограничение на неотрицательность X1 не нарушено.

Ограничение на неотрицательность X2 нарушено на 2,53623049360914E-05 единиц.

Ограничение на неотрицательность X3 не нарушено.

Ограничение на неотрицательность X4 не нарушено.

Ограничение на неотрицательность X5 не нарушено.

Ограничение на неотрицательность X6 не нарушено.

Ограничение на неотрицательность X7 не нарушено.

Ограничение на неотрицательность X8 не нарушено.

Штраф (сумма невязок): 3,02341472256802E-05

Время работы программы: 0:13.63364

Пример 9.

Итерация 8: 1,33427878779068 4,54270412099935 0,452617428124004 5,43093053754322 3,37925191003059 1,57112198376183 2,96413827981382 -0,000506451504840645 5,86470718565881

Расстояние до идеальной точки f\*: 141,114417144293

f\* = (266,207880034718; 287,799784017279)

f = (185,77337397766; 171,85337846139)

Ограничение 1 не нарушено.

Ограничение 2 не нарушено.

Ограничение 3 не нарушено.

Ограничение 4 не нарушено.

Ограничение 5 не нарушено.

Ограничение 6 нарушено на 7,77284234914077E-05 единиц.

Ограничение 7 не нарушено.

Ограничение 8 не нарушено.

Ограничение на неотрицательность X1 не нарушено.

Ограничение на неотрицательность X2 не нарушено.

Ограничение на неотрицательность X3 не нарушено.

Ограничение на неотрицательность X4 не нарушено.

Ограничение на неотрицательность X5 не нарушено.

Ограничение на неотрицательность X6 не нарушено.

Ограничение на неотрицательность X7 не нарушено.

Ограничение на неотрицательность X8 нарушено на 0,000506451504840645 единиц.

Ограничение на неотрицательность X9 не нарушено.

Штраф (сумма невязок): 0,000584179928332053

Время работы программы: 0:00.09472

Пример 10.

Итерация 3: 0,825972428773525 1,64632433135137 1,2172519359006 1,11118391538827 1,39397667543651 0,907604833062439 0,98837003219589 1,03550847946462 0,840480422370946 -2,36158638614848E-05

Расстояние до идеальной точки f\*: 145,08944876416

f\* = (141,831971995333; 181,10421277117)

f = (53,7000904723306; 65,8491929156186)

Ограничение 1 нарушено на 3,19307874008246E-06 единиц.

Ограничение 2 не нарушено.

Ограничение 3 не нарушено.

Ограничение 4 не нарушено.

Ограничение 5 не нарушено.

Ограничение 6 не нарушено.

Ограничение 7 не нарушено.

Ограничение 8 не нарушено.

Ограничение 9 не нарушено.

Ограничение на неотрицательность X1 не нарушено.

Ограничение на неотрицательность X2 не нарушено.

Ограничение на неотрицательность X3 не нарушено.

Ограничение на неотрицательность X4 не нарушено.

Ограничение на неотрицательность X5 не нарушено.

Ограничение на неотрицательность X6 не нарушено.

Ограничение на неотрицательность X7 не нарушено.

Ограничение на неотрицательность X8 не нарушено.

Ограничение на неотрицательность X9 не нарушено.

Ограничение на неотрицательность X10 нарушено на 2,36158638614848E-05 единиц.

Штраф (сумма невязок): 2,68089426015672E-05

Время работы программы: 0:00.45097

Пример 11.

Итерация 34: 0,0108890599600815 11,9496930092559 7,12138353607867 4,06519654554201

Расстояние до идеальной точки f\*: 49,403570858639

f\* = (113,764099037139; 141,544704264099; 201,796423658872)

f = (104,185512820596; 103,668878735733; 171,557618453944)

Ограничение 1 нарушено на 8,05821329663559E-08 единиц.

Ограничение 2 не нарушено.

Ограничение 3 не нарушено.

Ограничение 4 не нарушено.

Ограничение на неотрицательность X1 не нарушено.

Ограничение на неотрицательность X2 не нарушено.

Ограничение на неотрицательность X3 не нарушено.

Ограничение на неотрицательность X4 не нарушено.

Штраф (сумма невязок): 8,05821329663559E-08

Время работы программы: 0:00.00977

Пример 12.

Итерация 9: 3,4748755967747 2,44406880244101 -1,39273002743012E-05 1,10556851045514 1,59005873550608

Расстояние до идеальной точки f\*: 19,4407001360125

f\* = (36,3477622890682; 33,7762289068232; 73,0726338958181)

f = (39,4257358042539; 52,2326595405735; 67,7974970179982)

Ограничение 1 не нарушено.

Ограничение 2 не нарушено.

Ограничение 3 не нарушено.

Ограничение 4 не нарушено.

Ограничение на неотрицательность X1 не нарушено.

Ограничение на неотрицательность X2 не нарушено.

Ограничение на неотрицательность X3 нарушено на 1,39273002743012E-05 единиц.

Ограничение на неотрицательность X4 не нарушено.

Ограничение на неотрицательность X5 не нарушено.

Штраф (сумма невязок): 1,39273002743012E-05

Время работы программы: 0:00.00914

Пример 13.

Итерация 5: 4,38823211387879 0,659843640924902 1,00663729716431 2,64077746681359 7,92491571871769 1,93123752231501

Расстояние до идеальной точки f\*: 13,742938952354

f\* = (98,6666666666667; 62,6666666666667; 122,666666666667; 20)

f = (96,3191399331069; 68,0286398370422; 116,45978085826; 30,7741032835651)

Ограничение 1 не нарушено.

Ограничение 2 не нарушено.

Ограничение 3 нарушено на 1,85507681820241E-05 единиц.

Ограничение 4 не нарушено.

Ограничение 5 не нарушено.

Ограничение на неотрицательность X1 не нарушено.

Ограничение на неотрицательность X2 не нарушено.

Ограничение на неотрицательность X3 не нарушено.

Ограничение на неотрицательность X4 не нарушено.

Ограничение на неотрицательность X5 не нарушено.

Ограничение на неотрицательность X6 не нарушено.

Штраф (сумма невязок): 1,85507681820241E-05

Время работы программы: 0:00.03065

Пример 14.

Итерация 6: 0,922864259064548 2,4715041513659 -3,83561342962448E-05 0,292863490034716 4,2510280474528 5,04433967300283 3,6909582912576 0,360549128273521 2,549717361724

Расстояние до идеальной точки f\*: 42,7695666319725

f\* = (26,672228763667; 108,329750490608; 91,660370058873; 140,998833753855)

f = (60,0297093580484; 115,487433047662; 85,4879229938665; 115,955215885736)

Ограничение 1 не нарушено.

Ограничение 2 не нарушено.

Ограничение 3 не нарушено.

Ограничение 4 не нарушено.

Ограничение 5 не нарушено.

Ограничение 6 не нарушено.

Ограничение 7 не нарушено.

Ограничение 8 не нарушено.

Ограничение на неотрицательность X1 не нарушено.

Ограничение на неотрицательность X2 не нарушено.

Ограничение на неотрицательность X3 нарушено на 3,83561342962448E-05 единиц.

Ограничение на неотрицательность X4 не нарушено.

Ограничение на неотрицательность X5 не нарушено.

Ограничение на неотрицательность X6 не нарушено.

Ограничение на неотрицательность X7 не нарушено.

Ограничение на неотрицательность X8 не нарушено.

Ограничение на неотрицательность X9 не нарушено.

Штраф (сумма невязок): 3,83561342962448E-05

Время работы программы: 0:00.04876

На основе проведенных расчетов можно прийти к выводу, что решения задач при подборе значений параметров можно получить быстро с достаточной высокой точностью, т.е. опасения по поводу сложности получения достаточно точного решения в силу овражности целевой функций и неэффективности метода наискорейшего спуска для такого рода функций, не оправдались. Соответственно, метод внешних штрафных функций дает решение, довольно близкое к допустимой области.

# 2.3. Решение прикладной оптимизационной задачи

Решим одну задачу на составление оптимального плана производства.

Небольшое швейное предприятие занимается пошивом форменной одежды. В следующей таблице приведены характеристики производственного процесса.

Таблица 1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Материал | Затраты на единицу продукции | | | | Запасы материалов |
| Брюки | Юбки | Пиджаки | Платья |
| Ткань, | 1.4 | 0.9 | 1.1 | 1.8 | 2500 |
| Пуговицы, шт. | 2 | 1 | 6 | 0 | 1000 |
| Молнии, м | 0.1 | 0.2 | 0 | 0.4 | 500 |
| Затраты рабочего времени, человеко-часы | 1 | 0.9 | 1.75 | 1.2 |  |
| Прибыль от единицы продукции, тыс. руб. | 2,6 | 2,15 | 2,55 | 3,1 |  |

Предприятию необходимо составить такой план выпуска продукции, чтобы максимизировать общую прибыль и минимизировать трудовые затраты.

Составим математическую модель задачи:

– объемы выпуска каждого вида продукции в шт.

Воспользуемся разработанной программой:

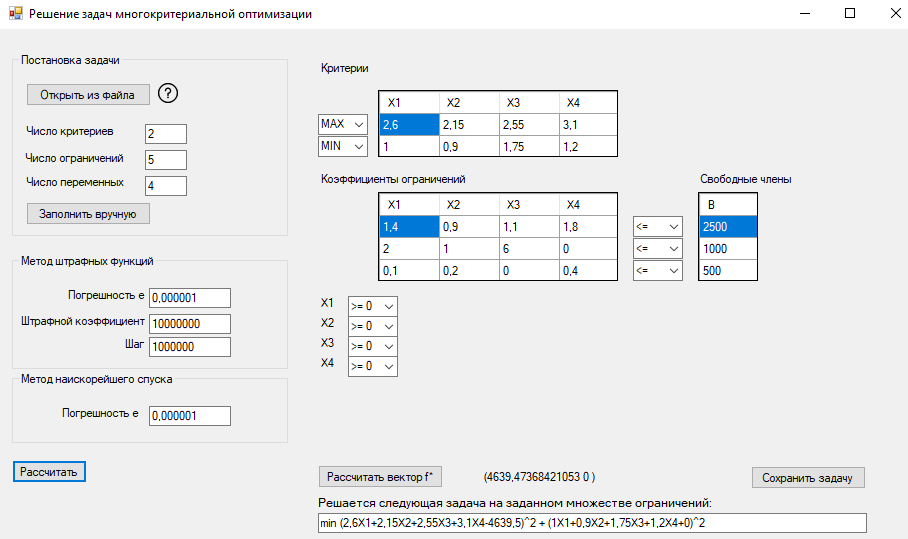


Рисунок 8 Постановка производственной задачи

Применяя симплексный метод, получим вектор оптимальных значений по двум частным критериям: f\* = (4639,47368421053; 0)

Сформулируем однокритериальную задачу.

Применим метод внешних штрафных функций с погрешностью 0.000001, начальным штрафным коэффициентом 10000000, шагом увеличения коэффициента 1000000 и погрешностью для метода наискорейшего спуска 0.000001.

# Итерация 1: 339,643988087873 320,717956899899 -0,00096044013487341 567,235955330751

# Итерация 2: 339,643988075066 320,717956898046 -0,000960600684243749 567,235955349964

# Расстояние до идеальной точки f\*: 1850,78013451872

# f\* = (4639,47368421053; 0)

# f = (3331,04698837911; 1308,97161465207)

# Ограничение 1 не нарушено.

# Ограничение 2 нарушено на 0,000169444072184888 единиц.

# Ограничение 3 не нарушено.

# Ограничение на неотрицательность X1 не нарушено.

# Ограничение на неотрицательность X2 не нарушено.

# Ограничение на неотрицательность X3 нарушено на 0,000960600684243749 единиц.

# Ограничение на неотрицательность X4 не нарушено.

# Штраф (сумма невязок): 0,00113004475642864

# Время работы программы: 0:00.36768

Проанализировав результаты, можно дать рекомендацию произвести 340 брюк, 320 юбок и 567 платьев. При таком объеме выпуска прибыль швейного предприятия будет составлять 3329.7 тысяч рублей, а затраты рабочего времени – 1308.4 человеко-часов.

# Заключение

Поставленная цель выпускной квалификационной работы, программно реализовать и численно исследовать метод на основе метрики в пространстве критериев для задач с линейными целевыми функциями и ограничениями, была достигнута.

На первом этапе работы был изучен и описан подход к решению многокритериальных задач на основе метрики в пространстве критериев. Изучены методы нелинейного программирования, выбран один вариант метода внешних штрафных функций для решения однокритериальной задачи, соответствующей исходной многокритериальной задачи. Среди методов безусловной минимизации для решения вспомогательных задач на каждой итерации метода штрафных функций был выбран метод наискорейшего спуска. Полный шаг находится методом дихотомии.

На втором этапе была разработана программа на языке C#, реализующая исследуемый метод, и спроектирован интерфейс к ней. Проведены численные эксперименты для задач различных размерностей, с разными видами целевых функций и областями ограничений. Была решена одна прикладная задача производственного типа. Программа показала успешность своей работы. Решения задач получены с достаточно высокой точностью довольно быстро.

В дальнейшем для исследования метода на основе метрики в пространстве критериев можно применить иные методы условной оптимизации и сравнить эффективность их работы; также можно расширить исследование на случаи с нелинейными частными критериями и ограничениями.

# Список используемой литературы

1. Васильев Ф. П. Численные методы решения экстремальных задач – Москва: Наука, 1988. – 552 с
2. Кашина О.А., Кораблёв А.И. Методы оптимизации. Часть II. Численные методы решения экстремальных задач, Казань, КГУ, 2011. - 144 с.
3. Карманов В.Г. Математическое программирование. - Москва.: Наука, 1975. – 260 с.
4. Ляшенко И.Н, Карагодова Е.А, Черникова Н.В., Шор Н.З. Линейное и нелинейное программирование. - Издательское объединение “Вища школа”, 2005. – 370 с.
5. Бродецкий Г.Л. Методы оптимизации многокритериальных решений в логистике. - Москва, 2009. – 220 c.
6. Jared L. Cohon. Multiobjective programming and planning. - Mineola, N.Y.: Dover Publications, 2003. – 333 p.
7. Andre A. Keller. Multi-Objective Optimization in Theory and Practice I: Classical Methods. - Bentham Science Publishers, Sharjah (UAE), 2017. – 296 p.

# Приложение. Листинг программы

Файл MulticriteriaProblem.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace MulticriteriaOptimization

{

public enum MathSign { GreaterThan, LessThan, Equal }

public class MultiCriteriaProblem

{

public bool[] Minimize { get; set; }

public double[,] CriteriaCoefficients { get; set; }

public double[,] ConstraintCoefficients { get; set; }

public double[] Constants { get; set; }

public MathSign[] ConstraintSigns { get; set; }

public int[] NotNonNegativeVarInd { get; set; }

public int CountCriteria { get; set; }

public int CountConstraint { get; set; }

public int CountVariables { get; set; }

public MultiCriteriaProblem(bool[] minimize, double[,] criteriaCoefficients, double[,] constraintCoefficients, double[] constants, MathSign[] constraintSigns, int[] notNonNegativeVariables)

{

Minimize = minimize;

ConstraintCoefficients = constraintCoefficients;

CriteriaCoefficients = criteriaCoefficients;

CountCriteria = criteriaCoefficients.GetLength(0);

CountConstraint = ConstraintCoefficients.GetLength(0);

CountVariables = ConstraintCoefficients.GetLength(1);

Constants = constants;

ConstraintSigns = constraintSigns;

NotNonNegativeVarInd = notNonNegativeVariables;

}

}

}

Файл SimplexMethod.cs

using System;

using System.Collections.Generic;

namespace MulticriteriaOptimization

{

class SimplexMethod

{

public MultiCriteriaProblem Prob { get; set; }

public int CriteriaInd { get; set; }

public double[,] SimplexTable { get; set; }

// Симплексная таблица:

// 0 строка - коэффициенты при переменных в целевой функции

// 1-n строки 1-m столбцы - коэффициенты при переменных в соответсвующих ограничениях

// n+1 строка - оценки

// n+2 строка - оценки: коэффициент при M

// 0 столбец - свободные члены

// Базис хранится отдельно

public int[] Basis { get; set; }

private int varInd;

private int costMRow;

private int costRow;

private const double M = double.PositiveInfinity;

private Dictionary<int, int> additionalVars;

public SimplexMethod(MultiCriteriaProblem prob, int criteriaInd)

{

Prob = prob;

CriteriaInd = criteriaInd;

SimplexTable = new double[Prob.CountConstraint + 3,

Prob.CountVariables + CountTotalNewVars() + 1];

varInd = Prob.CountVariables + 1;

costMRow = SimplexTable.GetLength(0) - 1;

costRow = costMRow - 1;

Basis = new int[Prob.CountConstraint];

additionalVars = new Dictionary<int, int>();

for (int i = 0; i < Prob.CountConstraint + 1; i++)

{

for (int j = 0; j < Prob.CountVariables + 1; j++)

{

if (i == 0 && j != 0)

{

SimplexTable[0, j] = prob.CriteriaCoefficients[CriteriaInd,j - 1];

}

else if (i != 0 && j == 0)

{

SimplexTable[i, 0] = prob.Constants[i - 1];

}

else if (i == 0 && j == 0)

{

continue;

}

else

{

SimplexTable[i, j] = prob.ConstraintCoefficients[i - 1, j - 1];

}

}

}

}

public double Calculate()

{

ConvertToStandardForm();

AddArtificialVariables();

FindInitialCosts();

int prev = -1;

while (true)

{

int pivotColumn = FindPivotColumn();

if (pivotColumn == -1 || pivotColumn == prev)

{

if (Math.Round(SimplexTable[SimplexTable.GetLength(0) - 1, 0], 10) != 0)

{

return double.NaN;

}

else

{

double[] solution = new double[SimplexTable.GetLength(1) - 1];

for (int i = 0; i < Basis.Length; i++)

{

if (Basis[i] - 1 < Prob.CountVariables)

{

solution[Basis[i] - 1] = SimplexTable[i + 1, 0];

}

}

if (additionalVars.Count != 0)

{

foreach (KeyValuePair<int, int> pair in additionalVars)

{

solution[pair.Key] -= solution[pair.Value];

}

}

double optObjectiveValue = SimplexTable[SimplexTable.GetLength(0) - 2, 0];

if (Prob.Minimize[CriteriaInd] == true)

{

optObjectiveValue \*= -1;

}

return optObjectiveValue;

}

}

else

{

int pivotRow = FindPivotRow(pivotColumn);

if (pivotRow == -1)

{

return double.NaN;

}

Basis[pivotRow - 1] = pivotColumn;

FindNewSimplexTable(pivotRow, pivotColumn);

prev = pivotColumn;

}

}

}

private int CountTotalNewVars()

{

int count = 0;

for (int i = 0; i < Prob.CountConstraint; i++)

{

if ((Prob.ConstraintSigns[i] == MathSign.LessThan && Prob.Constants[i] >= 0) ||

(Prob.ConstraintSigns[i] == MathSign.GreaterThan && Prob.Constants[i] < 0))

{

count++;

}

else if ((Prob.ConstraintSigns[i] == MathSign.GreaterThan && Prob.Constants[i] >= 0) ||

(Prob.ConstraintSigns[i] == MathSign.LessThan && Prob.Constants[i] < 0))

{

count += 2;

}

else if (Prob.ConstraintSigns[i] == MathSign.Equal)

{

count++;

}

}

if (Prob.NotNonNegativeVarInd != null)

{

count += Prob.NotNonNegativeVarInd.Length;

}

return count;

}

private void ConvertToStandardForm()

{

ToMax();

MakeConstantsPositive();

MakeAllVarNonNegative();

AddSlackAndSurplusVariables();

}

private void ToMax()

{

if (Prob.Minimize[CriteriaInd] == true)

{

for (int i = 1; i < Prob.CountVariables + 1; i++)

{

SimplexTable[0, i] \*= -1;

}

}

}

private void MakeConstantsPositive()

{

for (int i = 0; i < Prob.CountConstraint; i++)

{

if (Prob.Constants[i] < 0)

{

for (int j = 0; j < SimplexTable.GetLength(1); j++)

{

SimplexTable[i + 1, j] \*= -1;

}

}

}

}

private void MakeAllVarNonNegative()

{

if (Prob.NotNonNegativeVarInd != null)

{

foreach (int ind in Prob.NotNonNegativeVarInd)

{

for (int i = 0; i < Prob.CountConstraint + 1; i++)

{

SimplexTable[i, varInd] = SimplexTable[i, ind + 1] \* (-1);

}

additionalVars.Add(ind, varInd - 1);

varInd++;

}

}

}

private void AddSlackAndSurplusVariables()

{

for (int i = 0; i < Prob.CountConstraint; i++)

{

if ((Prob.ConstraintSigns[i] == MathSign.LessThan && Prob.Constants[i] >= 0) ||

(Prob.ConstraintSigns[i] == MathSign.GreaterThan && Prob.Constants[i] < 0))

{

SimplexTable[i + 1, varInd] = 1;

Basis[i] = varInd;

varInd++;

}

else if ((Prob.ConstraintSigns[i] == MathSign.GreaterThan && Prob.Constants[i] >= 0) ||

(Prob.ConstraintSigns[i] == MathSign.LessThan && Prob.Constants[i] < 0))

{

SimplexTable[i + 1, varInd] = -1;

varInd++;

}

}

}

private void AddArtificialVariables()

{

for (int i = 0; i < Prob.CountConstraint; i++)

{

if ((Prob.ConstraintSigns[i] == MathSign.GreaterThan && Prob.Constants[i] >= 0) ||

(Prob.ConstraintSigns[i] == MathSign.LessThan && Prob.Constants[i] < 0) ||

Prob.ConstraintSigns[i] == MathSign.Equal)

{

SimplexTable[i + 1, varInd] = 1;

SimplexTable[0, varInd] = -M;

Basis[i] = varInd;

varInd++;

}

}

}

private void FindInitialCosts()

{

int costMRow = SimplexTable.GetLength(0) - 1;

int costRow = costMRow - 1;

for (int j = 0; j < SimplexTable.GetLength(1); j++)

{

SimplexTable[costRow, j] = 0;

SimplexTable[costMRow, j] = 0;

for (int i = 1; i < SimplexTable.GetLength(0) - 2; i++)

{

if (SimplexTable[0, Basis[i - 1]] == -M)

{

SimplexTable[costMRow, j] += (SimplexTable[i, j] \* (-1));

}

else

{

SimplexTable[costRow, j] += (SimplexTable[i, j] \* SimplexTable[0, Basis[i - 1]]);

}

}

if (SimplexTable[0, j] == -M)

{

SimplexTable[costMRow, j] += 1;

}

else

{

SimplexTable[costRow, j] -= SimplexTable[0, j];

}

}

}

private void FindNewSimplexTable(int pivotRow, int pivotColumn)

{

double[,] oldTable = new double[SimplexTable.GetLength(0), SimplexTable.GetLength(1)];

Array.Copy(SimplexTable, oldTable, SimplexTable.Length);

double pivotElem = SimplexTable[pivotRow, pivotColumn];

for (int j = 0; j < SimplexTable.GetLength(1); j++)

{

SimplexTable[pivotRow, j] = oldTable[pivotRow, j] / pivotElem;

}

for (int i = 1; i < SimplexTable.GetLength(0); i++)

{

if (i == pivotRow || oldTable[i, pivotColumn] == 0)

continue;

for (int j = 0; j < SimplexTable.GetLength(1); j++)

{

SimplexTable[i, j] = oldTable[i, j] - (oldTable[i, pivotColumn] \* SimplexTable[pivotRow, j]);

}

}

}

private int FindPivotColumn()

{

List<int> minInd = new List<int>();

double minM = SimplexTable[costMRow, 1];

for (int j = 2; j < SimplexTable.GetLength(1); j++)

{

if(SimplexTable[costMRow, j] < minM)

{

minM = SimplexTable[costMRow, j];

}

}

for (int j = 1; j < SimplexTable.GetLength(1); j++)

{

if (SimplexTable[costMRow, j] == minM)

{

minInd.Add(j);

}

}

if(minInd.Count == 1)

{

return minInd[0];

}

else

{

double min = SimplexTable[costRow, 1];

int pivotCol = minInd[0];

foreach(int j in minInd)

{

if (SimplexTable[costRow, j] < min)

{

min = SimplexTable[costRow, j];

pivotCol = j;

}

}

if(min >= 0 && minM >=0)

{

return -1;

}

else

return pivotCol;

}

}

private int FindPivotRow(int pivotCol)

{

int pivotRow = -1;

for (int i = 1; i < SimplexTable.GetLength(0) - 2; i++)

{

if (SimplexTable[i, pivotCol] > 0)

{

pivotRow = i;

break;

}

}

if (pivotRow == -1)

return pivotRow;

for (int i = pivotRow + 1; i < SimplexTable.GetLength(0) - 2; i++)

{

if ((SimplexTable[i, pivotCol] > 0) && ((SimplexTable[i, 0] / SimplexTable[i, pivotCol]) <

(SimplexTable[pivotRow, 0] / SimplexTable[pivotRow, pivotCol])))

{

pivotRow = i;

}

}

return pivotRow;

}

}

}

Файл PenaltyMethod.cs

using System;

using System.Collections.Generic;

namespace MulticriteriaOptimization

{

public class PenaltyMethod

{

public MultiCriteriaProblem Prob { get; set; }

public double[] IdealF { get; }

public double AlphaK { get; set; }

double stepPenalty;

double epsilon;

double epsilonGrad;

List<double[]> gradDescentIterations;

public List<double[]> PenaltyIterations { get; set; }

public PenaltyMethod(MultiCriteriaProblem prob, double[] idealF, double epsilon, double epsilonGrad, double alphaK, double step)

{

Prob = prob;

IdealF = idealF;

this.epsilon = epsilon;

this.epsilonGrad = epsilonGrad;

AlphaK = alphaK;

stepPenalty = step;

PenaltyIterations = new List<double[]>();

gradDescentIterations = new List<double[]>();

}

public double[] Calculate()

{

double[] xk = new double[Prob.CountVariables];

for (int i = 0; i < xk.Length; i++)

{

xk[i] = 0;

}

double[] prev = new double[xk.Length];

double norm;

do

{

Array.Copy(xk, prev, xk.Length);

xk = DeepGradientDescent(xk);

PenaltyIterations.Add(xk);

AlphaK += stepPenalty;

norm = VectorNorm(SubstractVectors(prev, xk));

}

while (norm > epsilon);

return xk;

}

private double[] DeepGradientDescent(double[] x0)

{

gradDescentIterations.Clear();

double[] xk = new double[x0.Length];

Array.Copy(x0, xk, x0.Length);

for (int k = 0; ; k++)

{

gradDescentIterations.Add(xk);

double[] prev = new double[x0.Length];

Array.Copy(xk, prev, xk.Length);

double[] sk = GetDerivativeInXk(xk);

double[] interval = FindIntervalForBisectionMethod(xk, sk);

double step = BisectionMethod(interval[0], interval[1], xk, sk);

for (int i = 0; i < xk.Length; i++)

{

xk[i] = xk[i] - step \* sk[i];

}

double norm = VectorNorm(SubstractVectors(prev, xk));

if (norm < epsilonGrad)

{

break;

}

}

return xk;

}

public double VectorNorm(double[] x)

{

double res = 0;

for(int i = 0; i < x.Length; i++)

{

res += x[i] \* x[i];

}

return Math.Sqrt(res);

}

public double[] SubstractVectors(double[] prev, double[] next)

{

double[] res = new double[prev.Length];

for (int i = 0; i < prev.Length; i++)

{

res[i] = next[i] - prev[i];

}

return res;

}

private double[] FindIntervalForBisectionMethod(double[] xk, double[] sk)

{

double[] res = { 0, 0 };

double step = 0.5;

double[] x = new double[xk.Length];

Array.Copy(xk, x, xk.Length);

double funcPrev = GetFunctionValue(x) + GetPenaltyValue(x);

while (true)

{

for (int i = 0; i < x.Length; i++)

{

x[i] = xk[i] - step \* sk[i];

}

double funcX = GetFunctionValue(x) + GetPenaltyValue(x);

if (funcX >= funcPrev)

{

res[0] = step-1;

res[1] = step;

return res;

}

else

{

funcPrev = funcX;

step += 0.5;

}

}

}

private double BisectionMethod(double a0, double b0, double[] xk, double[] grad)

{

double lk, mk;

double epsilon = 0.000000000001;

double delta = 0.5 \* epsilon;

double ak = a0, bk = b0;

double[] x1 = new double[xk.Length];

double[] x2 = new double[xk.Length];

do

{

lk = (ak + bk - delta) / 2;

mk = (ak + bk + delta) / 2;

for (int i = 0; i < x1.Length; i++)

{

x1[i] = xk[i] - lk \* grad[i];

x2[i] = xk[i] - mk \* grad[i];

}

if (GetFunctionValue(x1) + GetPenaltyValue(x1) <= GetFunctionValue(x2) + GetPenaltyValue(x2))

{

bk = mk;

}

else

{

ak = lk;

}

} while ((bk - ak) >= epsilon);

return (ak + bk) / 2;

}

public double GetFunctionValue(double[] x)

{

double sum = 0;

double temp = 0;

for (int i = 0; i < Prob.CountCriteria; i++)

{

temp = 0;

for (int j = 0; j < Prob.CountVariables; j++)

{

temp += Prob.CriteriaCoefficients[i, j] \* x[j];

}

temp -= IdealF[i];

temp \*= temp;

sum += temp;

}

return sum;

}

public double[] GetDerivativeInXk(double[] x)

{

double[] sk = new double[x.Length];

double[] sum = new double[Prob.CountCriteria];

for (int i = 0; i < Prob.CountCriteria; i++)

{

for (int j = 0; j < Prob.CountVariables; j++)

{

sum[i] += Prob.CriteriaCoefficients[i,j] \* x[j];

}

sum[i] -= IdealF[i];

sum[i] \*= 2;

}

for (int i = 0; i < sk.Length; i++)

{

for (int j = 0; j < Prob.CountCriteria; j++)

{

sk[i] += sum[j] \* Prob.CriteriaCoefficients[j, i];

}

}

double[] sum2 = new double[Prob.CountConstraint];

for (int i = 0; i < Prob.CountConstraint; i++)

{

for (int j = 0; j < Prob.CountVariables; j++)

{

sum2[i] += Prob.ConstraintCoefficients[i, j] \* x[j];

}

sum2[i] -= Prob.Constants[i];

}

for (int i = 0; i < sk.Length; i++)

{

for (int j = 0; j < Prob.CountConstraint; j++)

{

if ((Prob.ConstraintSigns[j] == MathSign.LessThan) && (sum2[j] > 0) ||

(Prob.ConstraintSigns[j] == MathSign.GreaterThan) && (sum2[j] < 0) ||

Prob.ConstraintSigns[j] == MathSign.Equal)

{

sk[i] += 2 \* AlphaK \* Math.Abs(sum2[j] \* Prob.ConstraintCoefficients[j, i]);

}

}

}

for (int i = 0; i < Prob.CountVariables; i++)

{

if (!ContainsValue(Prob.NotNonNegativeVarInd, i) && x[i] < 0)

{

sk[i] += 2 \* AlphaK \* Math.Abs(x[i]);

}

}

return sk;

}

public double GetPenaltyValue(double[] x)

{

double sum = 0;

for (int i = 0; i < Prob.CountConstraint; i++)

{

double temp = 0;

for (int j = 0; j < Prob.CountVariables; j++)

{

temp += Prob.ConstraintCoefficients[i, j] \* x[j];

}

temp -= Prob.Constants[i];

if ((Prob.ConstraintSigns[i] == MathSign.LessThan) && (temp > 0) ||

(Prob.ConstraintSigns[i] == MathSign.GreaterThan) && (temp < 0) ||

Prob.ConstraintSigns[i] == MathSign.Equal)

{

sum += temp \* temp \* AlphaK;

}

}

for(int i = 0; i < Prob.CountVariables; i++)

{

if(!ContainsValue(Prob.NotNonNegativeVarInd,i) && x[i] < 0)

{

sum += AlphaK \* x[i] \* x[i];

}

}

return sum;

}

public bool ContainsValue(int[] arr, int val)

{

bool contains = false;

if (arr != null)

{

for (int i = 0; i < Prob.NotNonNegativeVarInd.Length; i++)

{

if (Prob.NotNonNegativeVarInd[i] == val)

{

contains = true;

break;

}

}

}

return contains;

}

public double CountDifferenceForConstraints(double[] x, int constrInd)

{

double sum = 0;

double temp = 0;

for (int j = 0; j < Prob.CountVariables; j++)

{

temp += Prob.ConstraintCoefficients[constrInd, j] \* x[j];

}

temp -= Prob.Constants[constrInd];

if ((Prob.ConstraintSigns[constrInd] == MathSign.LessThan) && (temp > 0) ||

(Prob.ConstraintSigns[constrInd] == MathSign.GreaterThan) && (temp < 0) ||

Prob.ConstraintSigns[constrInd] == MathSign.Equal)

{

sum += Math.Abs(temp);

}

return sum;

}

public double CountDifferenceForNonNegativityConstraints(double[] x, int constrInd)

{

double sum = 0;

if (!ContainsValue(Prob.NotNonNegativeVarInd, constrInd) && x[constrInd] < 0)

{

sum += Math.Abs(x[constrInd]);

}

return sum;

}

}

}

Файл MainForm.cs

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Windows.Forms;

using System.IO;

namespace MulticriteriaOptimization

{

public partial class MainForm : Form

{

int countCrit;

int countConstr;

int countVar;

MultiCriteriaProblem prob;

double[] solutionsF;

DataGridView criteriaTable;

DataGridView coeffTable;

DataGridView constTable;

string wrongFileFormatMessage = "Неверный формат данных в файле! Проверьте файл на соответствие правилам оформления исходных данных.";

public MainForm()

{

InitializeComponent();

textBoxCountConstraint.Text = "5";

textBoxCountCriteria.Text = "2";

textBoxCountVar.Text = "4";

textBoxAlpha.Text = "100";

textBoxEps.Text = "0,01";

textBoxEpsGrad.Text = "0,01";

textBoxStep.Text = "10";

}

private void MainForm\_Load(object sender, EventArgs e)

{

}

private void buttonCreateEmptyProblem\_Click(object sender, EventArgs e)

{

panel1.Controls.Clear();

prob = null;

countCrit = Convert.ToInt32(textBoxCountCriteria.Text);

countVar = Convert.ToInt32(textBoxCountVar.Text);

countConstr = Convert.ToInt32(textBoxCountConstraint.Text);

createUIProblem(countCrit, countVar, countConstr, false);

buttonComputeF.Visible = true;

buttonSaveProb.Visible = true;

labelOptF.Visible = false;

labelProblem.Visible = false;

textBoxProb.Visible = false;

}

private void createUIProblem(int countCrit, int countVar, int countConstr, bool generate)

{

Label criteriaLabel = new Label();

criteriaLabel.Location = new Point(0, 0);

criteriaLabel.Text = "Критерии";

criteriaLabel.AutoSize = true;

panel1.Controls.Add(criteriaLabel);

criteriaTable = new DataGridView();

criteriaTable.AllowUserToAddRows = false;

criteriaTable.Width = 60 \* countVar;

criteriaTable.RowHeadersVisible = false;

criteriaTable.Location = new Point(criteriaLabel.Location.X + 60, groupBox1.Location.Y + 5);

Random rand = new Random();

for (int i = 0; i < countVar; i++)

{

DataGridViewTextBoxColumn col = new DataGridViewTextBoxColumn();

col.Name = "X" + (i + 1);

col.Width = 60;

criteriaTable.Columns.Add(col);

}

criteriaTable.RowCount = countCrit;

int distance = 53;

for (int i = 0; i < countCrit; i++)

{

ComboBox min = new ComboBox();

min.Width = 50;

min.Items.Add("MIN");

min.Items.Add("MAX");

if(prob != null)

{

min.SelectedIndex = (prob.Minimize[i] == true) ? 0 : 1;

}

else if (generate == true)

{

min.SelectedIndex = (rand.NextDouble() >= 0.5) ? 0 : 1;

}

else

{

min.SelectedIndex = 0;

}

min.Location = new Point(criteriaLabel.Location.X, criteriaLabel.Location.Y + distance);

distance += 22;

panel1.Controls.Add(min);

}

int sum = criteriaTable.ColumnHeadersHeight;

foreach (DataGridViewRow row in criteriaTable.Rows)

sum += row.Height;

criteriaTable.Height = sum;

criteriaTable.ScrollBars = ScrollBars.None;

for(int i = 0; i < criteriaTable.RowCount; i++)

{

for (int j = 0; j < criteriaTable.ColumnCount; j++)

{

if(prob != null)

{

criteriaTable[j, i].Value = prob.CriteriaCoefficients[i, j];

}

else if (generate == true)

{

criteriaTable[j, i].Value = Math.Round(rand.NextDouble() \* 10,1);

}

else

{

criteriaTable[j, i].Value = 0;

}

}

}

panel1.Controls.Add(criteriaTable);

Label coeffLabel = new Label();

coeffLabel.Location = new Point(0, criteriaTable.Location.Y + criteriaTable.Height + 15);

coeffLabel.Text = "Коэффициенты ограничений";

coeffLabel.AutoSize = true;

panel1.Controls.Add(coeffLabel);

coeffTable = new DataGridView();

coeffTable.AllowUserToAddRows = false;

coeffTable.Width = 60 \* countVar;

coeffTable.RowHeadersVisible = false;

coeffTable.Location = new Point(coeffLabel.Location.X + 60, coeffLabel.Location.Y + 20);

for (int i = 0; i < countVar; i++)

{

DataGridViewTextBoxColumn col = new DataGridViewTextBoxColumn();

col.Name = "X" + (i + 1);

col.Width = 60;

coeffTable.Columns.Add(col);

}

coeffTable.RowCount = countConstr;

sum = coeffTable.ColumnHeadersHeight;

foreach (DataGridViewRow row in coeffTable.Rows)

sum += row.Height;

coeffTable.Height = sum;

coeffTable.ScrollBars = ScrollBars.None;

for (int i = 0; i < coeffTable.RowCount; i++)

{

for (int j = 0; j < coeffTable.ColumnCount; j++)

{

if (prob != null)

{

coeffTable[j, i].Value = prob.ConstraintCoefficients[i, j];

}

else if (generate == true)

{

coeffTable[j, i].Value = Math.Round(rand.NextDouble() \* 5,1);

}

else

{

coeffTable[j, i].Value = 0;

}

}

}

panel1.Controls.Add(coeffTable);

distance = 24;

ComboBox lessThan = new ComboBox();

for (int i = 0; i < countConstr; i++)

{

lessThan = new ComboBox();

lessThan.Width = 50;

lessThan.Items.Add("<=");

lessThan.Items.Add(">=");

lessThan.Items.Add("=");

if (prob != null)

{

switch(prob.ConstraintSigns[i])

{

case MathSign.LessThan:

lessThan.SelectedIndex = 0; break;

case MathSign.GreaterThan:

lessThan.SelectedIndex = 1; break;

case MathSign.Equal:

lessThan.SelectedIndex = 2; break;

}

}

else if (generate == true)

{

double r = rand.NextDouble();

if (r <= 0.7)

{

lessThan.SelectedIndex = 0;

}

else if (r > 0.7 && r < 0.99)

{

lessThan.SelectedIndex = 1;

}

else if(r >= 0.99) {

lessThan.SelectedIndex = 2;

}

}

else

{

lessThan.SelectedIndex = 0;

}

lessThan.Location = new Point(coeffTable.Location.X + coeffTable.Width + 15, coeffTable.Location.Y + distance);

distance += 22;

panel1.Controls.Add(lessThan);

}

Label constLabel = new Label();

constLabel.Location = new Point(lessThan.Location.X + lessThan.Width + 15, criteriaTable.Location.Y + criteriaTable.Height + 15);

constLabel.Text = "Свободные члены";

constLabel.AutoSize = true;

panel1.Controls.Add(constLabel);

constTable = new DataGridView();

constTable.AllowUserToAddRows = false;

constTable.Width = 60;

constTable.RowHeadersVisible = false;

constTable.Location = new Point(constLabel.Location.X, coeffTable.Location.Y);

constTable.ColumnCount = 1;

constTable.Columns[0].Name = "B";

constTable.RowCount = countConstr;

sum = constTable.ColumnHeadersHeight;

foreach (DataGridViewRow row in constTable.Rows)

sum += row.Height;

constTable.Height = sum;

constTable.ScrollBars = ScrollBars.None;

for (int i = 0; i < constTable.RowCount; i++)

{

if (prob != null)

{

constTable[0, i].Value = prob.Constants[i];

}

else if (generate == true)

{

constTable[0, i].Value = Math.Round(rand.NextDouble() \* 150);

}

else

{

constTable[0, i].Value = 0;

}

}

panel1.Controls.Add(constTable);

distance = 15;

for (int i = 0; i < countVar; i++)

{

Label X = new Label();

X.Location = new Point(criteriaLabel.Location.X, coeffTable.Location.Y + coeffTable.Height + distance);

X.Text = "X" + (i + 1);

X.AutoSize = true;

panel1.Controls.Add(X);

ComboBox zero = new ComboBox();

zero.Width = 50;

zero.Location = new Point(X.Location.X + X.Width + 10, coeffTable.Location.Y + coeffTable.Height + distance);

zero.Items.Add(">= 0");

zero.Items.Add(" ");

if (prob != null && prob.NotNonNegativeVarInd != null && prob.NotNonNegativeVarInd.Contains(i))

{

zero.SelectedIndex = 1;

}

else

{

zero.SelectedIndex = 0;

}

panel1.Controls.Add(zero);

distance += 20;

}

}

private void buttonOpenFromFile\_Click(object sender, EventArgs e)

{

prob = null;

panel1.Controls.Clear();

labelOptF.Visible = false;

labelProblem.Visible = false;

textBoxProb.Visible = false;

OpenFileDialog dlg = new OpenFileDialog();

dlg.DefaultExt = ".txt";

dlg.Filter = "Текстовый документ (.txt)| \*.txt";

if (dlg.ShowDialog() == DialogResult.OK)

{

string filename = dlg.FileName;

string extension = Path.GetExtension(filename);

try

{

if (extension == ".txt")

{

string[] fileLines = File.ReadAllLines(filename);

if (fileLines.Length == 0)

{

throw new Exception("Пустой файл!");

}

string[] split = fileLines[0].Split(' ');

if (!Int32.TryParse(split[0], out countCrit))

throw new Exception(wrongFileFormatMessage);

if (!Int32.TryParse(split[1], out countConstr))

throw new Exception(wrongFileFormatMessage);

if (!Int32.TryParse(split[2], out countVar))

throw new Exception(wrongFileFormatMessage);

bool[] minimize = new bool[countCrit];

double[,] criteriaCoefficients = new double[countCrit,countVar];

double[,] constraintCoefficients = new double[countConstr, countVar];

double[] constants = new double[countConstr];

MathSign[] constraintSigns = new MathSign[countConstr];

int[] NotNonNegativeVarInd;

for (int i = 1; i <= countCrit; i++)

{

string line = fileLines[i];

string[] splitStr = line.Split(' ');

minimize[i - 1] = splitStr[0] == "1";

for (int j = 1; j <= countVar; j++)

{

if (!Double.TryParse(splitStr[j], out criteriaCoefficients[i - 1, j - 1]))

throw new Exception(wrongFileFormatMessage);

}

}

for (int i = countCrit + 1; i < countConstr + countCrit + 1; i++)

{

string line = fileLines[i];

string[] splitStr = line.Split(' ');

for (int j = 0; j < countVar; j++)

{

if (!Double.TryParse(splitStr[j], out constraintCoefficients[i - countCrit - 1, j]))

throw new Exception(wrongFileFormatMessage);

}

if (!Double.TryParse(splitStr[countVar + 1], out constants[i - countCrit - 1]))

throw new Exception(wrongFileFormatMessage);

switch(splitStr[countVar])

{

case "<=": constraintSigns[i - countCrit - 1] = MathSign.LessThan; break;

case ">=": constraintSigns[i - countCrit - 1] = MathSign.GreaterThan; break;

case "=": constraintSigns[i - countCrit - 1] = MathSign.Equal; break;

default: throw new Exception(wrongFileFormatMessage);

}

}

if(fileLines.Length == countConstr + countCrit + 2)

{

string[] splitStr = fileLines[countConstr + countCrit + 1].Split(' ');

NotNonNegativeVarInd = new int[splitStr.Length];

for(int i = 0; i < splitStr.Length; i++)

{

if (!Int32.TryParse(splitStr[i], out NotNonNegativeVarInd[i]))

throw new Exception(wrongFileFormatMessage);

}

}

else

{

NotNonNegativeVarInd = null;

}

try

{

prob = new MultiCriteriaProblem(minimize, criteriaCoefficients, constraintCoefficients, constants, constraintSigns, NotNonNegativeVarInd);

createUIProblem(countCrit, countVar, countConstr, false);

}

catch (Exception exc)

{

MessageBox.Show(exc.Message);

}

}

buttonComputeF.Visible = true;

buttonSaveProb.Visible = true;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

private void buttonCompute\_Click(object sender, EventArgs e)

{

bool[] minimize = new bool[countCrit];

double[,] criteriaCoefficients = new double[countCrit, countVar];

double[,] constraintCoefficients = new double[countConstr, countVar];

double[] constants = new double[countConstr];

MathSign[] constraintSigns = new MathSign[countConstr];

int[] NotNonNegativeVarInd;

List<string> comboboxText = new List<string>();

foreach(Control c in panel1.Controls)

{

if(c is ComboBox)

comboboxText.Add(((ComboBox)c).SelectedItem.ToString());

}

for(int i = 0; i < countCrit; i++)

{

minimize[i] = comboboxText[i] == "MIN";

}

for (int i = countCrit; i < countCrit + countConstr; i++)

{

switch(comboboxText[i])

{

case "<=": constraintSigns[i - countCrit] = MathSign.LessThan; break;

case ">=": constraintSigns[i - countCrit] = MathSign.GreaterThan; break;

case "=": constraintSigns[i - countCrit] = MathSign.Equal; break;

}

}

List<int> tempNotNonNeg = new List<int>();

for (int i = countCrit + countConstr; i < countCrit + countConstr + countVar; i++)

{

if(comboboxText[i] == " ")

{

tempNotNonNeg.Add(i - countCrit - countConstr);

}

}

NotNonNegativeVarInd = tempNotNonNeg.ToArray();

for (int i = 0; i < criteriaTable.RowCount; i++)

{

for (int j = 0; j < criteriaTable.ColumnCount; j++)

{

criteriaCoefficients[i,j] = Convert.ToDouble(criteriaTable[j, i].Value);

}

}

for (int i = 0; i < coeffTable.RowCount; i++)

{

for (int j = 0; j < coeffTable.ColumnCount; j++)

{

constraintCoefficients[i, j] = Convert.ToDouble(coeffTable[j, i].Value);

}

}

for (int i = 0; i < constTable.RowCount; i++)

{

constants[i] = Convert.ToDouble(constTable[0, i].Value);

}

prob = new MultiCriteriaProblem(minimize, criteriaCoefficients, constraintCoefficients, constants, constraintSigns, NotNonNegativeVarInd);

solutionsF = new double[prob.Minimize.Length];

SimplexMethod sm;

labelOptF.Text = "(";

for (int i = 0; i < prob.Minimize.Length; i++)

{

sm = new SimplexMethod(prob, i);

solutionsF[i] = sm.Calculate();

labelOptF.Text += solutionsF[i] + " ";

}

labelOptF.Text += ")";

labelOptF.Visible = true;

labelProblem.Text = "Решается следующая задача на заданном множестве ограничений:";

textBoxProb.Text = "min ";

for (int i = 0; i < prob.CriteriaCoefficients.GetLength(0); i++)

{

textBoxProb.Text += "(";

for (int j = 0; j < prob.CriteriaCoefficients.GetLength(1); j++)

{

textBoxProb.Text += prob.CriteriaCoefficients[i,j] + "X" + (j+1);

if (j != prob.CriteriaCoefficients.GetLength(1) - 1)

{

textBoxProb.Text += (prob.CriteriaCoefficients[i, j + 1] >= 0) ? "+" : "";

}

}

textBoxProb.Text += (solutionsF[i] > 0) ? "-" : "+";

textBoxProb.Text += Math.Abs(Math.Round(solutionsF[i],1));

textBoxProb.Text += ")^2";

if(i != prob.CriteriaCoefficients.GetLength(0) - 1)

{

textBoxProb.Text += " + ";

}

}

labelProblem.Visible = true;

textBoxProb.Visible = true;

buttonComputePenalty.Visible = true;

}

private void buttonComputePenalty\_Click(object sender, EventArgs e)

{

try

{

double eps, epsGrad, a, step;

if (!Double.TryParse(textBoxEps.Text, out eps))

throw new Exception("Неверный формат погрешности для метода штрафных функций");

if (!Double.TryParse(textBoxEpsGrad.Text, out epsGrad))

throw new Exception("Неверный формат погрешности для метода наискорейшего спуска");

if (!Double.TryParse(textBoxAlpha.Text, out a))

throw new Exception("Неверный формат штрафного коэффициента");

if (!Double.TryParse(textBoxStep.Text, out step))

throw new Exception("Неверный формат шага");

if (Double.IsNaN(solutionsF[0]))

throw new Exception("Решаются только задачи с совместной системой граничений и огранниченными сверху/снизу функциями");

PenaltyMethod pm = new PenaltyMethod(prob, solutionsF, eps, epsGrad, a, step);

//Stopwatch sw = Stopwatch.StartNew();

//double[] x = pm.Calculate();

//sw.Stop();

ResultsForm res = new ResultsForm(pm);

res.Show();

}

catch(Exception exc)

{

MessageBox.Show(exc.Message);

}

}

private void buttonGenerate\_Click(object sender, EventArgs e)

{

panel1.Controls.Clear();

prob = null;

countCrit = Convert.ToInt32(textBoxCountCriteria.Text);

countVar = Convert.ToInt32(textBoxCountVar.Text);

countConstr = Convert.ToInt32(textBoxCountConstraint.Text);

createUIProblem(countCrit, countVar, countConstr, true);

buttonComputeF.Visible = true;

buttonSaveProb.Visible = true;

labelOptF.Visible = false;

labelProblem.Visible = false;

textBoxProb.Visible = false;

}

private void buttonSaveProb\_Click(object sender, EventArgs e)

{

try

{

SaveFileDialog saveFileDialog1 = new SaveFileDialog();

saveFileDialog1.Filter = "Текстовый документ(.txt)| \*.txt";

saveFileDialog1.RestoreDirectory = true;

saveFileDialog1.FilterIndex = 1;

if (saveFileDialog1.ShowDialog() == DialogResult.OK)

{

string filename = saveFileDialog1.FileName;

string extension = Path.GetExtension(filename);

if (extension == ".txt")

{

using (StreamWriter sw = new StreamWriter(filename))

{

sw.Write(countCrit + " " + countConstr + " " + countVar);

sw.WriteLine();

List<string> comboboxText = new List<string>();

foreach (Control c in panel1.Controls)

{

if (c is ComboBox)

comboboxText.Add(((ComboBox)c).SelectedItem.ToString());

}

for (int i = 0; i < countCrit; i++)

{

sw.Write((comboboxText[i] == "MIN")?"1 ":"0 ");

for (int j = 0; j < countVar; j++)

{

sw.Write(Convert.ToDouble(criteriaTable[j, i].Value));

if (j != countVar - 1) sw.Write(" ");

}

sw.WriteLine();

}

int k = countCrit;

for (int i = 0; i < countConstr; i++)

{

for (int j = 0; j < countVar; j++)

{

sw.Write(Convert.ToDouble(coeffTable[j, i].Value) + " ");

}

sw.Write(comboboxText[k] + " ");

k++;

sw.Write(Convert.ToDouble(constTable[0, i].Value));

sw.WriteLine();

}

}

}

else

throw new Exception("Неверное расширение файла!");

}

//MessageBox.Show("Задача успешно сохранена!");

}

catch (Exception exc)

{

MessageBox.Show(exc.Message);

}

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

SaveFileDialog saveFileDialog1 = new SaveFileDialog();

saveFileDialog1.Filter = "Текстовый документ(.txt)| \*.txt";

saveFileDialog1.RestoreDirectory = true;

saveFileDialog1.FilterIndex = 1;

if (saveFileDialog1.ShowDialog() == DialogResult.OK)

{

string filename = saveFileDialog1.FileName;

string extension = Path.GetExtension(filename);

if (extension == ".txt")

{

using (StreamWriter sw = new StreamWriter(filename))

{

List<string> comboboxText = new List<string>();

foreach (Control c in panel1.Controls)

{

if (c is ComboBox)

comboboxText.Add(((ComboBox)c).SelectedItem.ToString());

}

for (int i = 0; i < countCrit; i++)

{

sw.Write((comboboxText[i] == "MIN") ? "min " : "max ");

for (int j = 0; j < countVar; j++)

{

if(Convert.ToDouble(criteriaTable[j, i].Value) != 0)

{

sw.Write(Convert.ToDouble(criteriaTable[j, i].Value) + "x" + (j + 1) + " ");

if (j != countVar - 1) sw.Write("+ ");

}

}

sw.WriteLine();

}

int k = countCrit;

for (int i = 0; i < countConstr; i++)

{

for (int j = 0; j < countVar; j++)

{

if (Convert.ToDouble(coeffTable[j, i].Value) != 0)

{

sw.Write(Convert.ToDouble(coeffTable[j, i].Value) + "x" + (j + 1) + " ");

}

if (j != countVar - 1) sw.Write("+ ");

}

sw.Write(comboboxText[k] + " ");

k++;

sw.Write(Convert.ToDouble(constTable[0, i].Value));

sw.WriteLine();

}

}

}

else

throw new Exception("Неверное расширение файла!");

}

}

catch (Exception exc)

{

MessageBox.Show(exc.Message);

}

}

}

}

Файл MainForm.cs

using System;

using System.Windows.Forms;

using System.Diagnostics;

using System.IO;

using Excel = Microsoft.Office.Interop.Excel;

namespace MulticriteriaOptimization

{

public partial class ResultsForm : Form

{

PenaltyMethod penalty;

double[] xOpt;

Stopwatch sw;

public ResultsForm(PenaltyMethod pm)

{

InitializeComponent();

this.penalty = pm;

sw = Stopwatch.StartNew();

xOpt = pm.Calculate();

sw.Stop();

WriteResults();

label1.Text = "Время работы программы: " + sw.Elapsed.ToString(@"m\:ss\.ff");

}

private void WriteResults()

{

for (int i = 0; i < penalty.PenaltyIterations.Count; i++)

{

textBox1.Text += "Итерация " + (i + 1) + ": ";

for (int j = 0; j < penalty.PenaltyIterations[i].Length; j++)

{

textBox1.Text += penalty.PenaltyIterations[i][j] + " ";

}

textBox1.Text += "\r\n";

}

penalty.AlphaK = 1;

textBox1.Text += "Расстояние до идеальной точки f\*: " + (Math.Sqrt(penalty.GetFunctionValue(xOpt)))+ "\r\n";

textBox1.Text += "f\* = (";

for (int i = 0; i < penalty.Prob.CountCriteria; i++)

{

textBox1.Text += penalty.IdealF[i];

if (i != penalty.Prob.CountCriteria - 1)

{

textBox1.Text += "; ";

}

}

textBox1.Text += ") \r\n";

double[] fOpt = new double[penalty.Prob.CriteriaCoefficients.GetLength(0)];

textBox1.Text += "f = (";

for (int i = 0; i < penalty.Prob.CountCriteria; i++)

{

double temp = 0;

for (int j = 0; j < penalty.Prob.CountVariables; j++)

{

temp += penalty.Prob.CriteriaCoefficients[i, j] \* xOpt[j];

}

textBox1.Text += temp;

if (i != penalty.Prob.CountCriteria-1)

{

textBox1.Text += "; ";

}

}

textBox1.Text += ") \r\n";

double totalSum = 0;

for (int i = 0; i < penalty.Prob.CountConstraint; i++)

{

double penForConstr = penalty.CountDifferenceForConstraints(xOpt,i);

totalSum += penForConstr;

textBox1.Text += (penForConstr!=0)? "Ограничение " + (i+1) + " нарушено на " + penForConstr + " единиц. \r\n":"Ограничение "+(i+1)+ " не нарушено. \r\n";

}

for (int i = 0; i < penalty.Prob.CountVariables; i++)

{

double penForNonNeg = penalty.CountDifferenceForNonNegativityConstraints(xOpt, i);

totalSum += penForNonNeg;

if (!penalty.ContainsValue(penalty.Prob.NotNonNegativeVarInd, i))

{

textBox1.Text += (penForNonNeg != 0) ? "Ограничение на неотрицательность X" + (i + 1) + " нарушено на " + penForNonNeg + " единиц. \r\n" : "Ограничение на неотрицательность X" + (i + 1) + " не нарушено. \r\n";

}

}

textBox1.Text += "Штраф (сумма невязок): " + totalSum + "\r\n";

textBox1.Text += "Время работы программы: " + sw.Elapsed.ToString(@"m\:ss\.fffff") + "\r\n";

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

SaveFileDialog saveFileDialog1 = new SaveFileDialog();

saveFileDialog1.Filter = "Текстовый документ(.txt)| \*.txt| Excel - файл(.xlsx) | \*.xlsx";

saveFileDialog1.RestoreDirectory = true;

saveFileDialog1.FilterIndex = 1;

if (saveFileDialog1.ShowDialog() == DialogResult.OK)

{

string filename = saveFileDialog1.FileName;

string extension = Path.GetExtension(filename);

if (extension == ".txt")

{

using (StreamWriter sw = new StreamWriter(filename))

{

sw.Write(textBox1.Text);

}

}

else if (extension == ".xlsx")

{

Excel.Application xlApp = new Excel.Application();

xlApp.Visible = false;

Excel.Workbook xlWorkbook = xlApp.Workbooks.Add(Excel.XlWBATemplate.xlWBATWorksheet);

Excel.Worksheet xlWorksheet = (Excel.Worksheet)xlWorkbook.Worksheets[1];

for (int i = 0; i < penalty.PenaltyIterations.Count; i++)

{

for (int j = 0; j < penalty.PenaltyIterations[i].Length; j++)

{

xlWorksheet.Cells[i + 1, j + 1] = penalty.PenaltyIterations[i][j];

}

}

xlWorkbook.SaveAs(filename);

GC.Collect();

GC.WaitForPendingFinalizers();

xlWorkbook.Close();

xlApp.Quit();

}

else

throw new Exception("Неверное расширение файла!");

}

}

catch (Exception exc)

{

MessageBox.Show(exc.Message);

}

}

}

}